Unjamming Lightning: A Systematic Approach

Clara Shikhelman and Sergei Tikhomirov

Chaincode Labs
{clara.shikhelman,sergey.s.tikhomirov}@gmail.com

Abstract. Users of decentralized financial networks suffer from inventive security exploits. Identity-based fraud prevention methods are inapplicable in these networks, as they contradict their privacy-minded design philosophy. Novel mitigation strategies are therefore needed. Their rollout, however, may damage other desirable network properties.
In this work, we introduce an evaluation framework for mitigation strategies in decentralized financial networks. This framework allows researchers and developers to examine and compare proposed protocol modifications along multiple axes, such as privacy, security, and user experience.
As an example, we focus on the jamming attack in the Lightning Network. Lightning is a peer-to-peer payment channel network on top of Bitcoin. Jamming is a cheap denial-of-service attack that allows an adversary to temporarily disable Lightning channels by flooding them with failing payments.
We propose a practical solution to jamming that combines unconditional fees and peer reputation. Guided by the framework, we show that, while discouraging jamming, our solution keeps the protocol incentive compatible. It also preserves security, privacy, and user experience, and is straightforward to implement. We support our claims analytically and with simulations. Moreover, our anti-jamming solution may help alleviate other Lightning issues, such as malicious channel balance probing.

Keywords: Bitcoin · Payment Channels · Lightning Network · Jamming

1 Introduction

Decentralized blockchain protocols, such as Bitcoin, represent a new paradigm for financial networks. Their core design principles are permissionless access[1] and user privacy. To that end, users are identified by public keys and can generate as many of those as they wish. Anyone can join the network without formal identification. This feature renders traditional identity-based anti-fraud methods inapplicable on the protocol level.

At the same time, decentralized financial networks are a valuable target for attackers. Users and service providers lose access to their funds or have their privacy eroded.

[1] Permissioned networks, in contrast, are run by a known set of participants. Such systems are outside the scope of this paper.

Reputation and economic incentives become the main tools for designing mitigation strategies in this setting. The former aims to selectively block misbehaving actors by distinguishing them from honest users. The latter impose a cost on undesired behavior, ideally making it prohibitively expensive, while potentially also compensating the victims.

When suggesting a protocol change, one should assess its effects on the network as a whole. The proposal must not introduce new attack vectors, diminish user experience, or jeopardize privacy and security. A failure to acknowledge and navigate inevitable trade-offs may render the mitigation strategy useless. Implementation difficulty should also be taken into account.

This work focuses on mitigation strategies against attacks on permissionless financial networks. Our major concern is that a proposed solution, while addressing the attack, would significantly damage other aspects of the protocol. To avoid "throwing out the baby with the bathwater", we introduce a general framework for the design and evaluation of mitigation strategies. As a case study, we consider mitigations of the jamming attack in the Lightning Network (LN).

The LN, a payment network on top of Bitcoin, has been designed to address an inherent limit on Bitcoin's transaction throughput. *Jamming* is a long-standing, and yet unfixed, denial-of-service attack in Lightning. It allows an attacker to cheaply and efficiently block victim channels. Jamming deprives LN users of the network's core functionality and reduces their routing fee revenue. Many approaches to mitigating jamming have been discussed in the LN community, but none has been implemented.

To demonstrate the power of our framework, we use it to distill an effective solution to jamming without any fundamental drawbacks.

Our Contributions

– We introduce a **general framework for evaluating attack mitigations** in decentralized financial networks. This framework helps ensure that suggested changes do not significantly diminish existing protocol guarantees.
– We propose a **mitigation strategy against jamming in Lightning**. Our solution combines unconditional fees and local reputation based on prior behavior. We justify our design choices by evaluating our solution through the lens of the framework defined above.

The rest of this paper is structured as follows. First, we introduce the evaluation framework (Section 2). Then we provide the background on the LN and on jamming in particular (Section 3). We apply the framework to the design decisions related to jamming mitigation strategies (Section 4), and introduce our solution to jamming (Section 5) that combines unconditional fees (Section 5.1) and local reputation (Section 5.2). We then discuss simulation results (Section 6) and review related work (Section 7). Finally, we outline future work directions (Section 8) and conclude (Section 9).

2 A Framework for Mitigation Evaluation

We suggest evaluating attack mitigations by the following criteria.

Effectiveness. A mitigation should discourage or prevent the attack. The adversary must bear a cost, which may take the form of money (fees or penalties), time (the need to accumulate reputation before attacking), or a combination thereof. Note that the attackers' motivation is not always purely financial (e.g., griefing).

Incentive Compatibility. A mitigation should keep the protocol incentive compatible. Decentralized networks rely on the rationality of their participants: it should be in their best interest to adhere to the rules. Payment networks should incentivize nodes to forward payments and to report errors truthfully.

User Experience (UX). A mitigation should not deteriorate the UX. Changes in user interface should be intuitive and easily explainable. UX should be evaluated from the viewpoint of end users as well as professional service providers.

Privacy & Security. A mitigation should not significantly diminish users' privacy. When it comes to security, we must ensure that a proposed protocol change does not introduce new attack vectors. Negative effects of a proposed mitigation should be carefully weighed against its potential benefits.

Ease of Implementation. A mitigation should be straightforward to implement. Protocol changes require rough consensus among developers [42]. Easily implementable proposals are more likely to be adopted and not perpetually delayed.

The framework outlined above can be used as follows. First, consider all possible classes of mitigation strategies. Second, exclude strategies that are clearly incompatible with at least one of the outlined criteria. Finally, compare the costs and benefits of the remaining options in the context of a particular network.

We note that the former two criteria (effectiveness and incentive compatibility) are required for a functioning solution. The latter three criteria often imply trade-offs. User experience can sometimes be improved by using a more centralized architecture or accumulating more user data, which erodes privacy. Similarly, implementation may be simpler for strategies that require users to additionally trust some designated third parties.

Note that some criteria listed above are (at least to some extent) objectively quantifiable, while others are largely subjective. Navigating these trade-offs depends on the network and the attack in question.

3 Overview of Lightning and Jamming

Permissionless blockchain networks suffer from inherent scalability challenges that second-layer (L2) protocols aim to solve [17]. They increase transaction throughput while leveraging the security guarantees of the underlying network.

The LN is a major Bitcoin-based L2 protocol [34]. Pairs of LN nodes open *payment channels* by locking up funds in a collaboratively owned address. They then make payments by reflecting the up-to-date distribution of funds in the *channel state*. Payments happen *off-chain* and are only settled *on-chain* when needed. A penalty mechanism ensures economic security of balance updates.

To initiate a *multi-hop* payment, the sender first finds a suitable route of channels to the receiver. Consider a node U_i on a route $U_0, \dots, U_m$, $0 \leq i \leq m$. The portion of the route towards the sender (U_j, where $0 \leq j < i$) is referred to as *upstream*. Analogously, the part of the route towards the receiver (U_j, where $i < j \leq m$) is called *downstream*. The peers of U_i on the route (U_{i-1} and U_{i+1}) are referred to as its upstream and downstream peer, respectively.

Each *routing node*, upon receiving a forwarding request, either *forwards* the payment or *fails* it. If a routing node fails a payment, it notifies the sender, who may then route around the erring node. The receiver, upon getting the payment, can either *claim* or fail it. By claiming the payment from the last routing node, the receiver reveals the secret that allows that node to claim the money from its upstream peer, and so on.

The receiving node cannot reject an incoming payment without updating the channel state with its upstream peer. Nodes are only supposed to fail a payment if they lack the resources (e.g., liquidity) to forward it. Nodes generally do not limit the amount of liquidity an incoming payment may use.

We call a payment *resolved* if it has either been claimed or has failed. Payments that are not resolved within a certain timeout are canceled and trigger channel closure[2].

Unresolved payments lock up liquidity in all channels along the route. This liquidity cannot be used to forward other payments. Moreover, each channel can only hold up to a certain number of unresolved payments at any one time. This limit stems from Bitcoin protocol rules: a fraudulent closure of a channel with too many unresolved payments cannot be disputed on-chain[3]. We say that each channel has a limited number of *payment slots*. Each unresolved payment occupies one slot and a portion of the liquidity.

LN payments are onion-routed for privacy. A routing node only knows its immediate peers on the route, but does not know the original sender or receiver[4].

Jamming

Jamming is a denial-of-service attack on LN channels. The attacker, who controls both ends of a target route, sends payments (*jams*) and delays claiming them, blocking liquidity and slots along the whole route. The attacker can fail jams and immediately send new ones, prolonging the attack. The goal of a jammer may be to disrupt a specific set of channels (e.g., belonging to a business competitor) or the whole network.

[2] See e.g. [8] for a comprehensive protocol description.

[3] At any given time, each channel direction may hold up to 483 unresolved payments. Users may set lower limits for their channels.

[4] This assumption can be violated with timing attacks [38].

We distinguish *liquidity-based* and *slot-based* jamming. The former locks up the liquidity of the victim channels, whereas the latter occupies all their slots. In general, liquidity-based jams have higher value than slot-based jams. To occupy a slot, a jam just needs to exceed the minimum payment size that nodes on the route agree to forward[5]. Slot-based jamming is arguably more efficient, as it only requires a constant amount of capital to block channels of any capacity.

We also differentiate between *quick* and *slow* jamming. In quick jamming, the attacker sends a continuous flow of jams that resolve in seconds, mimicking failed honest payments. In contrast, slow jams are resolved after a very long delay (on the order of hours or days), which allows the victim to detect the attack. The borderline between quick and slow jamming depends on a subjective definition of the maximal honest payment resolution delay.

The attacker may pursue different goals. We note that users join the network for two main reasons: to send and receive payments and to earn fees from routing. Thus, the jammer may want to prevent users from exchanging payments or to deprive routing service providers of fee revenue.

The jamming attack is essentially free. The jammer pays no fees, as routing nodes do not charge for failed[6] payment attempts. The only expense for the attacker is the opportunity cost of capital[7] locked up for the duration of jamming. Onion routing further complicates jamming mitigation.

4 Design Decisions for a Jamming Mitigation

Multiple mitigation strategies against jamming have been proposed [14,31]. They can be grouped into monetary and reputation-based. To systematically analyze the solution space, we start by listing the design decisions to be made. We then evaluate our options based on the framework introduced in Section 2.

4.1 Monetary Strategies

Monetary mitigation strategies aim to make attacks expensive. These strategies can also compensate the victim for the damage. In the context of jamming, the monetary approach implies charging fees for failed payment attempts in addition to existing (*success-case*) fees. We consider the following design decisions.

– **Who receives the fee?** The receivers may be: the downstream peer, an agreed-upon third party, or no one (provably burning the fee).

[5] Slot-based jams must also be larger than the *dust limit* – the minimal amount that occupies a slot in a forwarding channel. Payments below the dust limit are allowed (albeit with weaker security guarantees), but they cannot be used in (slot-based) jamming because they do not occupy slots. Due to their minuscule value, they are impractical for liquidity-based jamming too.

[6] Technically, an attacker may also resolve the jams as regular successful payments, which would incur paying fees. In what follows, we assume that jams always fail.

[7] For further details on channel costs, see [16].

– **What currency is the fee paid in?** Fees might be paid in the native asset of the network (bitcoins in the case of the LN) or in another asset. In schemes that imply fee burning, fees can take the form of a solution to a proof-of-work puzzle.
– **Is the fee conditional?** The simplest fee design is an unconditional fee paid *upfront*, that is, while offering a payment. An alternative approach would be to return the (fail-case) upfront fee if the payment succeeds. Other conditions may also be considered.
– **How is the fee amount calculated?** Currently, LN fees are composed of a base fee and a part that is linearly proportional to the payment amount. More complex formulas may include other parameters besides the amount.

Paying a fee to a peer is simpler than to a third party, as introducing a third party complicates the incentives. The LN offers no obvious way to provably burn funds[8], which makes proof-of-burn schemes difficult to implement. Hence, we narrow down our decision space to monetary fees paid to the downstream peer. Non-refundable unconditional fees are easier to implement than fees with refunds. As for the fee amount calculation, for simplicity, we adopt the existing success-case fee structure (i.e., a base fee and a proportional part). We assume that fee amounts are always positive[9] (i.e., paid downstream, not upstream). Making the fee amount dependent of the factual payment resolution time would be desirable, yet we are currently unaware of any reliable way to implement this idea. When it comes to UX, privacy, and security, the available design choices share similar benefits and drawbacks.

4.2 Reputation-Based Strategies

In reputation-based strategies, nodes track their level of trust towards other nodes. In the jamming context, reputation scores help routing nodes distinguish good peers from bad peers, and then either fail payments offered by the latter or disconnect from them altogether. When designing a reputation scheme, we make the following design decisions.

– **Whose reputation affects payment forwarding?** A common design pattern in P2P networks [26] implies that a routing node just considers its upstream peer's reputation. Alternatively, the original sender's reputation may be attached to the payment. We call a reputation scheme *local* if each node only assigns reputation scores to its peers, and *global* otherwise.
– **Does reputation require consensus?** Nodes can assign reputation scores independently or try to reach consensus about the reputation of all nodes in the network.

[8] In contrast, bitcoins on the base layer can be sent to a provably unspendable address.
[9] Bi-directional fees that include a component paid upstream have been discussed [14]. As advertising a negative fee may attract unwanted traffic, this proposal needs further evaluation.

- **Is reputation fungible?** Fungible reputation takes the form of tokens that can be transferred between nodes. Non-fungible reputation, in contrast, is inseparable from the node is was initially assigned to.
- **How is reputation assigned?** One way to assign reputation scores is to do so based on the peer's prior behavior. Another approach implies assigning reputation based on commitments to some scarce resource, such as proof-of-work solutions or proofs of bitcoin ownership (*stake certificates* [32]).

Attaching the sender's reputation to payments contradicts the LN's privacy-focused goals. As nodes can only objectively perceive the actions of their peers, reputation should be local. In light of Sybil attacks, we opt for independent rather than consensus-based reputation. We prefer non-fungible schemes, as secondary markets for reputation tokens are non-trivial to design securely[10]. Finally, we choose to assign reputation scores based on prior behavior rather than commitments to a scarce resource. Proof-of-work does not look promising in this respect, judging from its failure as a spam prevention technique [23]. In particular, puzzle difficulty that is sufficient to deter attacks turned out to be unacceptably high for honest users. We leave the evaluation of other resource-based reputation schemes for future work.

5 Our Solution to Jamming

Based on the arguments laid out in Section 4, we propose a combination of two strategies to mitigate jamming.

1. **Unconditional fee** paid to the downstream peer addresses quick jamming by imposing a small cost on every payment.
2. **Local reputation** based on past behavior addresses slow jamming by punishing peers who forward payments that take too long to resolve.

In Subsections 5.1 and 5.2, we give further details on unconditional fees and local reputation, respectively. We examine both parts of our mitigation strategy through the lens on the framework introduced in Section 2 and discuss best practices for parameter choices.

5.1 Unconditional Fee

Consider a payment route $U_0, \dots, U_m$. Let $f_{i,i+1}$ denote a fee that U_i pays to U_{i+1}, where $i \in [0, m-1]$. Let us denote success-case fees as $f^{\mathcal{S}}$ and unconditional fees as $f^{\mathcal{N}}$. For $X \in \{\mathcal{S}, \mathcal{N}\}$, the *fee revenue* of type X is the difference between what U_i pays and what it receives:

$$F_i^X = f_{i-1,i}^X - f_{i,i+1}^X \tag{1}$$

For a routing node U_i, the proposed fee scheme is as follows (see Figure 1).

[10] Prior research has shown that designing a secondary market for utility tokens is a challenging task [10,13,45]. For instance, large entities could hoard large quantities of such tokens to manipulate the market.

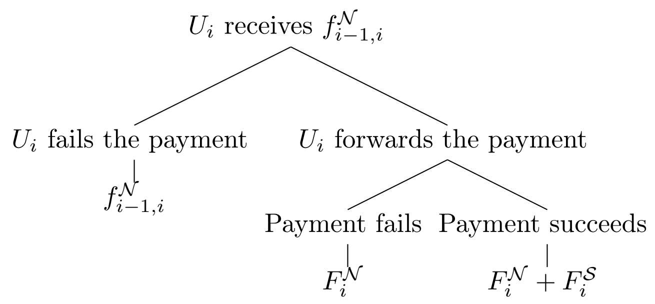
U_i receives $f^{\mathcal{N}}_{i-1,i}$
U_i fails the payment
U_i forwards the payment
$f^{\mathcal{N}}_{i-1,i}$
Payment fails
Payment succeeds
$F^{\mathcal{N}}_i$
$F^{\mathcal{N}}_i + F^{\mathcal{S}}_i$

Fig. 1. Decision tree for a routing node U_i. The values at leaves denote total fee revenue.

1. U_{i-1} pays $f^{\mathcal{N}}_{i-1,i}$ to U_i.
2. U_i decides either to fail or to forward the payment. If U_i fails the payment, no more fees are paid.
3. If U_i decides to forward the payment, it pays $f^{\mathcal{N}}_{i,i+1}$ to U_{i+1}.
4. The process continues until the payment either fails downstream or succeeds. If the payment succeeds, U_i receives $f^{\mathcal{S}}_{i-1,i}$ and pays $f^{\mathcal{S}}_{i,i+1}$.

The fee revenue for U_i is equal to $f^{\mathcal{N}}_{i-1,i}$ if U_i fails the payment without forwarding, $F^{\mathcal{N}}_i$ if the payment is forwarded and fails, or $F^{\mathcal{S}}_i + F^{\mathcal{N}}_i$ if it succeeds.

Note that each fee payment f includes not only the fee for the neighboring peer, but for all routing nodes further downstream. As an example, consider a four-node route (U_1, U_2, U_3, U_4), where U_2 and U_3 charge a flat routing fee of 1 satoshi[11] per payment. The sender U_1 would attach a fee of 2 satoshis when forwarding a payment to U_2: $f_{1,2} = 2$, but $f_{2,3} = 1$. Fee revenues would then be equal to $F_2 = f_{1,2} - f_{2,3} = 2 - 1 = 1$ and $F_3 = f_{2,3} - f_{3,4} = 1 - 0 = 1$. This applies to both success-case and unconditional fees.

Effectiveness With unconditional fees, jamming is no longer free. Moreover, the two components of the unconditional fee (a base fee and a proportional part) address slot-based and liquidity-based jamming, respectively. Low-value jams aimed at occupying slots are discouraged with base fees, whereas the proportional component plays a larger role in deterring high-value liquidity-targeted jams.

With properly set fee coefficients, fees may provide a similar revenue to routing nodes compared to what they earn from honest payments[12], compensating them for the financial damage of jamming. Unconditional fees can be relatively low to achieve this effect, as the attacker continuously sends jams to keep channels blocked, while honest payments usually only occupy a small portion of channel resources. Simulations confirm this intuition (see Section 6).

[11] The smallest unit of bitcoin. 1 bitcoin equals 100 million satoshis.
[12] Honest fee revenue of a given node depends on its position in the network, its liquidity management practices (e.g., rebalancing), and other factors.

Incentive Compatibility To forward a payment, a routing node allocates a slot and a portion of liquidity in one of its channels with its downstream peer. These resources remain unavailable until the payment succeeds or fails, which becomes costly if the payment takes long to resolve. Hence, we face an incentive compatibility challenge: a routing node can take the unconditional fee and then deliberately fail the payment. To ensure that nodes are incentivized to forward if possible, success-case fees must compensate them for the associated risk.

Let θ be the probability of payment failure. If U_i decides to forward the payment, its expected fee revenue is:

$$\begin{aligned}\mathbb{E}(F_i|Forward) =&(1-\theta)(F_i^{\mathcal{S}} + F_i^{\mathcal{N}}) + \theta F_i^{\mathcal{N}} \qquad (2)\\ =&F_i^{\mathcal{N}} + (1-\theta)F_i^{\mathcal{S}} = (f_{i-1,i}^{\mathcal{N}} - f_{i,i+1}^{\mathcal{N}}) + (1-\theta)F^{\mathcal{S}}. \qquad (3)\end{aligned}$$

The fee revenue if U_i decides to fail the payment is simply $f_{i-1,i}^{\mathcal{N}}$.

To ensure that U_i prefers to forward the payment rather than to fail it, the expected revenue for forwarding must be higher than for failing:

$$f_{i-1,i}^{\mathcal{N}} - f_{i,i+1}^{\mathcal{N}} + (1-\theta)F^{\mathcal{S}} > f_{i-1,i}^{\mathcal{N}} \qquad (4)$$

$$(1-\theta)F^{\mathcal{S}} > f_{i,i+1}^{\mathcal{N}} \qquad (5)$$

In other words, the expected *additional* revenue from forwarding must at least compensate the routing node for the unconditional fee it would pay downstream.

User Experience The key UX concern is that the fees for failed payment attempts would deter users. Wallets can feasibly abstract this detail away, as the expected number of attempts per payment is low[13] even if the payment failure probability θ is moderately high. For the payment to succeed with probability at least p after at most K attempts, the following needs to hold:

$$1 - \theta^K > p \qquad (6)$$

This is equivalent to:

$$\begin{aligned}\log(1-p) &> K\log\theta \qquad (7)\\ &\Updownarrow \; as \;\; \log\theta < 0\\ \frac{\log(1-p)}{\log\theta} &< K. \qquad (8)\end{aligned}$$

The number of required attempts grows slowly (logarithmically in required success probability p). Even assuming $\theta = 20\%$, a single attempt gives a success probability of $p = 80\%$, two attempts guarantee $p = 96\%$, and three attempts

[13] Assuming wallets retry payments, which is not always possible on mobile devices.

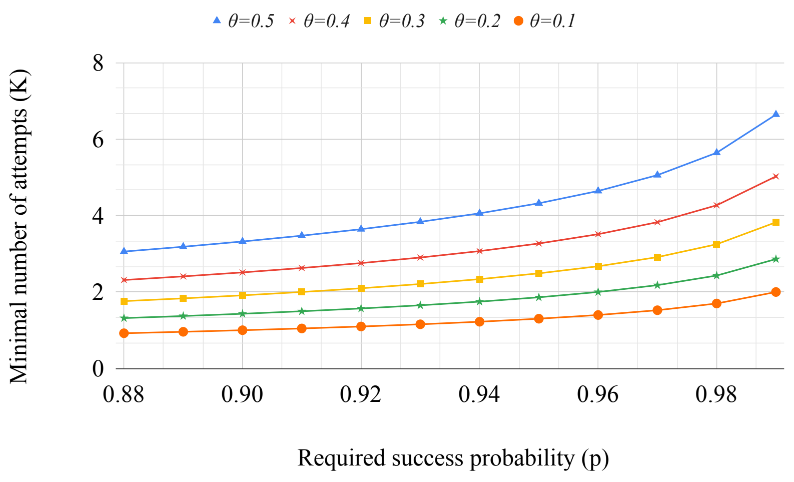

Fig. 2. The number of payment attempts needed as a function of the required success probability p, for various probabilities of payment failures θ.

suffice for $p = 99\%$. We conclude that the negative UX effects would be minimal due to a small number of attempts per payment, as well as low unconditional fee amounts (see Figure 2).

Privacy and Security Routing nodes should not know their position within the route. Unconditional fees weaken this claim, as they allow a routing node to deduce its distance to the receiver from fee amounts and public fee policies[14]. This issue can be addressed by the sender allocating a part of the final amount (what the receiver is supposed to get) to the last-hop unconditional fee, as if the route extended beyond the receiver. The drawback of this approach is that high unconditional fees lead to misaligned incentives for routing nodes. We do not expect this issue to be too serious, considering low unconditional fee amounts.

Implementation Unconditional fees are straightforward to implement. One approach could be to pay unconditional fees upfront. In an alternative method, a downstream node U_i withholds the fee $f^{\mathcal{N}}_{i-1,i}$ when returning the payment amount to U_{i-1} in case of failure, as a proof-of-concept implementation demonstrates [43]. Other implementation tasks include advertising extended fee policies in gossip messages and accounting for unconditional fees in route selection.

[14] This issue does not apply to the same extent to success-case fees because they are merged into the body of the payment. Routing nodes do not know which portion of the amount represents downstream success-case fees (they can make educated guesses). Unconditional fees, in contrast, are separate from the payment body.

5.2 Local Reputation

A reputation scheme is defined by three components: initialization, reputation updates, and making decisions based on reputation scores.

Initialization In the initialization phase, a node sets the following parameters:

- τ (seconds) – the maximal resolution time to consider a payment honest;
- t (seconds), T (seconds), A (satoshis per second) – reputation update parameters (see below);
- K (integer), L (satoshis) – the high-risk quota of slots and liquidity.

The values for the parameters defined above depend on the node's risk preferences. For instance, lower values for K and L reflect a higher risk tolerance. Routing nodes should weigh the potentially increased revenue from honest but high-risk payments (e.g., from new nodes) against jamming risk. The values of t, T, and A may depend on the cost of opening new channels that would handle honest payments while existing channels remain jammed.

Updating Reputation Scores We consider two possible values for reputation scores: *high* and *low* (more granular schemes are possible). Initially, all peers receive a low score. A payment is deemed *honest* if it resolves within τ seconds. Otherwise, it is considered a jam. A peer's behavior is defined as *good* if it only offers honest payments for forwarding. Moreover, those payments must pay at least A satoshis per second in fees to the routing node. Reputation of a peer grows if it demonstrates good behavior for a long enough period.

More precisely, a reputation score is updated as follows:

- to *high*, if the peer has maintained good behavior for duration t;
- to *low*, if there was no interval of good behavior of duration t within a time window of duration T.

A reputation update algorithm based on a sliding time window forgives occasional mistakes while punishing consistent misbehavior. This is a common pattern in peer-to-peer reputation systems.

If peers agree beforehand to tolerate payments with intentionally delayed resolution[15], such payments do not harm the offering node's reputation.

Forwarding The *offering* node may (but does not have to) mark a payment as low-risk, thereby *endorsing* it. Its downstream (*receiving*) peer considers a payment as *low-risk* if and only if it is endorsed by a high-reputation peer. Low-risk payments are forwarded on the best effort basis. High-risk payments, in contrast, can only use the quota of K slots and L satoshis of liquidity. The high-risk quota is defined per channel and not per peer. This prevents an attacker from gaining an advantage by opening many channels to the victim.

[15] This is useful for some L2 protocols, such as atomic swaps [48] (e.g., submarine swaps [6]) and Discreet Log Contracts [24]. More research is needed to analyze such protocols in a multi-hop setting.

Effectiveness A low-reputation attacker can no longer fully jam a channel that allocates at least some resources exclusively to low-risk payments. To circumvent this countermeasure, the attacker would need to build up reputation in advance, which requires effort and resources. Recall that in order to be assigned a high reputation score, a node must forward payments that pay A satoshis per second in fees for at least t seconds.

The downside of the proposed reputation scheme is potential under-utilization of channel resources. In case of high payment flow, some honest high-risk payments may fail due to the lack of resources in the high-risk quota.

Incentive Compatibility Routing nodes should be motivated to endorse payments to the best of their knowledge. The two possible deviating strategies are endorsing a high-risk payment and not endorsing a low-risk one. Not endorsing a low-risk payment clearly decreases the node's own fee revenue. More interestingly, by endorsing a high-risk payment, a node increases its expected revenue if the payment succeeds at the risk of losing reputation with its downstream peer if it turns out to be a jam. By setting reputation parameters appropriately, this strategy can be made unprofitable. If regaining reputation is very expensive, additional fee revenue would not justify falsely endorsing high-risk payments.

User Experience The UX consequences of our proposal largely concern new users, who can experience higher failure rates due to their initially low reputation. For casual users, this may not be a significant issue, as high-risk quotas in their peers' channels would be sufficient to handle their low payment volume. Moreover, professional LN service providers may offer more permissive reputation policies to their clients.

Privacy and Security By choosing whether to endorse a payment, a node leaks information on its possible origin. To preserve the sender's privacy, nodes may choose not to endorse some low-risk payments (at the cost of fee revenue).

A security risk for a reputation scheme is a cascading attack, where jams sent through long routes decrease reputation scores network-wide. This would harm the overall network performance, as nodes would wrongfully fail many low-risk payments. Even though decreasing routing nodes' reputation makes payments more likely to fail, it cannot stop the payment flow completely.

Cascading network dynamics have been studied in various contexts, such as finance [30], IoT [15], and viral spread [47]. These papers have quantified the risks based on theoretical results (such as [7,11,35]). In light of these studies and the LN properties [27], we conjecture that cascading attacks would be difficult. Further research on the parameter choice and attack strategies is needed.

Implementation The implementation of the proposed reputation system is rather straightforward. Additional fields should be added to the payment data structure to encode endorsement. Nodes should also be able to record their peers' reputation scores in their local databases.

6 Simulation

We implement an LN simulator that models payments with unconditional fees. Let the success-case fee be calculated as:

$$f^{\mathcal{S}} = b^{\mathcal{S}} + r^{\mathcal{S}} a$$

where $b^{\mathcal{S}}$ is the success-case base fee, $r^{\mathcal{S}}$ is the success-case proportional fee rate, and a is the payment amount.

Let n be the *unconditional fee coefficient*, which is the ratio between the unconditional fee and the success-case fee. That is, $b^{\mathcal{N}} = nb^{\mathcal{S}}$ and $r^{\mathcal{N}} = nr^{\mathcal{S}}$ are the unconditional base fee and proportional fee, respectively. The unconditional fee structure is analogous to the success-case fee:

$$f^{\mathcal{N}} = b^{\mathcal{N}} + r^{\mathcal{N}} a$$

The goal of our simulations is to deduce the *breakeven point* $\hat{n}$ – the minimal value of n that provides routing nodes with the same revenue under jamming as they earn from honest payments.

We make the following assumptions:

– honest payment amounts are distributed lognormally [25] with a mean of 50 000 satoshis[16] and $\sigma = 0.7$;
– for a channel with capacity c, a payment with amount a fails[17] with probability $\min\{1, a/c\}$;
– an honest resolution time is 1 second plus an exponentially distributed random value with expectation of 3 seconds.

Each jam has the amount of 354 satoshis (the dust limit) and resolves after 7 seconds. Each channel has 483 slots[18].

We consider two configurations: a channel-based one and a node-based one. In the channel-based configuration, we forward payments through a chain of three channels. The parties of the channel in the middle are considered routing nodes (they do not send their own payments). We vary the routing channel capacity from 0.1 to 10 million satoshis. For the node-based configuration, we consider a medium-sized LN node from a public network snapshot[19]. This routing node has five bidirectional channels with approximate capacities (in million satoshis): 0.3, 1, 1.001, 1.021, and 1.1. The node does not send payments, it only forwards payments from one of its peers to another. We model the rest of the network as one "virtual" node connected to all other nodes involved.

For both configurations, we run simulations in two scenarios. In the honest scenario, routing nodes forwards honest payments with a frequency varying from

[16] An equivalent of 10 USD at 20 thousand USD per bitcoin.
[17] We do not model balance distributions in channels. We do, however, model channel capacities, payment amounts, and fees calculated based on them.
[18] The LN specifications define the maximum number of slots [3] and the minimum payments size [4].
[19] Node ID `0263a6d2f0fed7b1e14d01a0c6a6a1c0fae6e0907c0ac415574091e7839a00405b`.

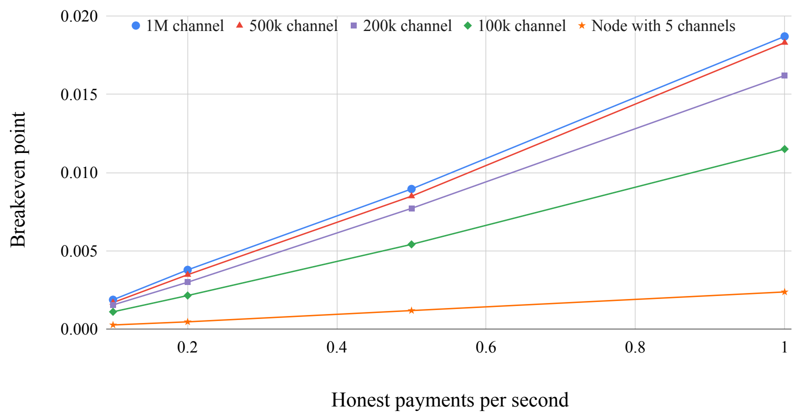

Fig. 3. The breakeven point (the minimal sufficient $\hat{n}$) for the node-based and various channel-based configurations, and for various honest payment frequencies.

0.1 to 1 payments per second. The success-case fee is set to 1 satoshi plus 5 parts per million. In the attack scenario, an attacker jams all channels with as few jams as possible (it may use looped routes in the node-base configuration). For each configuration, we iterate through values of n to find $\hat{n}$.

We observe that $\hat{n}$ is quite low for all considered scenarios (Figure 3). For a 1 million satoshi channel that routes 1 honest payment per second, the breakeven point $\hat{n} = 1.88\%$. In other words, our jamming countermeasure only increases the total fee by 1.88% by introducing an unconditional component. For a smaller channel with a capacity of 100 thousand satoshis, $\hat{n}$ is even lower (1.15%).

As we increase channel capacity, $\hat{n}$ goes up, but reaches saturation after 500 thousand satoshis. The explanation is that smaller channels do not earn that much from honest payments, as more of them fail. This effect, in turn, decreases the unconditional fees needed to offset the lost revenue. For larger channel capacities, the difference in capacity has no strong effect on $\hat{n}$, as most payments succeed.

For the node-based configuration, $\hat{n}$ is lower than for the channel-based configuration. This is explained by the fact that the attacker must forward more jams through the victim node to jam all of its channels. The honest payment flow, however, does not depend on the configuration. As a result, for a node-based configuration that implies more jams, a lower fee per jam is sufficient to offset honest revenue.

We conclude that even a small unconditional fee effectively compensates routing nodes for the damage from jamming.

7 Related Work

Jamming discussions date back to as early as 2015 [1,40]. Summaries of modern approaches are provided in [14,31,36].

The impact of jamming, as well as attack optimizations and potential countermeasures, has been widely studied [28,29,33,37,46]. Upfront payments have been previously discussed as a countermeasure [41,20]. Related proposals include stake certificates [32] and node-level firewall-like defenses [2]. Centralized global node scoring is available but rarely used in practice [5].

Attacks like flood-and-loot [18] and mass exit [44] exploit L2 protocols by inducing base-layer congestion. Privacy attacks on the LN include balance probing [12,19], timing attacks [38], and cross-layer deanonymization [21,39]. No mitigation strategies have been adopted to date.

8 Future Work

Future work directions include simulating more attack scenarios and reputation-based defense strategies. More broadly, feasibility of alternative design choices can be considered. Accounting for payment resolution times in fee amounts and privacy-preserving sender reputation are of particular interest, although these ideas present practical and theoretical challenges yet to be resolved. Additionally, the cost and efficiency of jamming can be studied for various attack goals and success metrics.

An effective countermeasure against jamming may alleviate other pressing LN issues. First, it would discourage probing attacks [19], which, similarly to jamming, require sending many failing payments. Second, our fee scheme may also be useful for trustless incentivization of watchtowers [9,22] – third-party services that dispute fraudulent channel closures on their users' behalf.

9 Conclusion

In this work, we have established an evaluation framework for attack mitigation strategies in decentralized financial networks. As a case study, we consider jamming – a long-standing denial-of-service attack vector threatening the Lightning Network, a prominent layer-two protocol on top of Bitcoin. After considering multiple design decisions through the lens of our framework, we propose an effective solution that mitigates jamming through a combination of unconditional fees and behavior-based local reputation. We demonstrate the feasibility of our proposal with simulations and analytical calculations.

Acknowledgments We thank Sergi Delgado Segura for the PoC implementation of unconditional fees. We thank Carla Kirk-Cohen, Joost Jager, and the LN developer community for thoughtful discussions.

References

1. Payment channel congestion via spam-attack. https://github.com/lightning/bolts/issues/182, 2017. 15
2. Circuit breaker. https://github.com/lightningequipment/circuitbreaker, 2020. 15
3. Bolt 02: Peer protocol. https://github.com/lightning/bolts/blob/master/02-peer-protocol.md, 2022. 13
4. Bolt 03: Transactions. https://github.com/lightningequipment/circuitbreaker, 2022. 13
5. The bos score, a lightning network node ranking. https://fulmo.org/bos-score.html, 2022. 15
6. Understanding submarine swaps. https://docs.lightning.engineering/the-lightning-network/multihop-payments/understanding-submarine-swaps, 2022. 11
7. Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *nature*, 406(6794):378–382, 2000. 12
8. Andreas M Antonopoulos, Olaoluwa Osuntokun, and René Pickhardt. *Mastering the Lightning Network*. " O'Reilly Media, Inc.", 2021. 4
9. Zeta Avarikioti, Orfeas Stefanos Thyfronitis Litos, and Roger Wattenhofer. Cerberus channels: Incentivizing watchtowers for bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 346–366. Springer, 2020. 15
10. Edmond Baranes, Ulrich Hege, and Jin-Hyuk Kim. Utility tokens financing, investment incentives, and regulation. *Investment Incentives, and Regulation (August 1, 2021)*, 2021. 7
11. Alain Barrat, Marc Barthelemy, and Alessandro Vespignani. *Dynamical processes on complex networks*. Cambridge university press, 2008. 12
12. Alex Biryukov, Gleb Naumenko, and Sergei Tikhomirov. Analysis and probing of parallel channels in the lightning network. *Cryptology ePrint Archive*, 2021. 15
13. Oliver Davey and Antti Jokinen. The money view vs. the accounting view: The capitol hill babysitting cooperative revisited. *Available at SSRN 3628707*, 2020. 7
14. Bastien Teinturier et al. Spamming the lightning network. https://github.com/t-bast/lightning-docs/blob/master/spam-prevention.md, 2022. 5, 6, 15
15. Xiuwen Fu, Pasquale Pace, Gianluca Aloi, Wenfeng Li, and Giancarlo Fortino. Cascade failures analysis of internet of things under global/local routing mode. *IEEE Sensors Journal*, 22(2):1705–1719, 2021. 12
16. Paolo Guasoni, Gur Huberman, and Clara Shikhelman. Lightning network economics: Channels. *Available at SSRN 3840374*, 2021. 5
17. Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 201–226. Springer, 2020. 3
18. Jona Harris and Aviv Zohar. Flood & loot: A systemic attack on the lightning network. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 202–213, 2020. 15
19. Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, Alejandro Ranchal-Pedrosa, Cristina Pérez-Solà, and Joaquin Garcia-Alfaro. On the difficulty of hiding the balance of lightning network channels. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 602–612, 2019. 15
20. Joost Jager. A proposal for up-front payments. https://lists.linuxfoundation.org/pipermail/lightning-dev/2020-March/002585.html, 2020. 15

21. George Kappos, Haaroon Yousaf, Ania Piotrowska, Sanket Kanjalkar, Sergi Delgado-Segura, Andrew Miller, and Sarah Meiklejohn. An empirical analysis of privacy in the lightning network. In *International Conference on Financial Cryptography and Data Security*, pages 167–186. Springer, 2021. 15
22. Majid Khabbazian, Tejaswi Nadahalli, and Roger Wattenhofer. Outpost: A responsive lightweight watchtower. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 31–40, 2019. 15
23. Ben Laurie and Richard Clayton. Proof-of-work proves not to work; version 0.2. In *Workshop on economics and information, security*, 2004. 7
24. Thibaut Le Guilly, Nadav Kohen, and Ichiro Kuwahara. Bitcoin oracle contracts: Discreet log contracts in practice. In *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–8. IEEE, 2022. 11
25. Gottfried Leibbrandt. A billion here, a billion there: The statistics of payments, 2009. 13
26. Sergio Marti and Hector Garcia-Molina. Limited reputation sharing in p2p systems. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 91–101, 2004. 6
27. Stefano Martinazzi and Andrea Flori. The evolving topology of the lightning network: Centralization, efficiency, robustness, synchronization, and anonymity. *Plos one*, 15(1):e0225966, 2020. 12
28. Subhra Mazumdar, Prabal Banerjee, Abhinandan Sinha, Sushmita Ruj, and Bimal Roy. Strategic analysis to defend against griefing attack in lightning network. *CoRR*, abs/2203.10533, 2022. 15
29. Ayelet Mizrahi and Aviv Zohar. Congestion attacks in payment channel networks. In *Financial Cryptography (2)*, volume 12675 of *Lecture Notes in Computer Science*, pages 170–188. Springer, 2021. 15
30. Akira Namatame and Takanori Komatsu. Management of systemic risks and cascade failures in a networked society. *Inf. Knowl. Syst. Manag.*, 10(1-4):111–133, 2011. 12
31. Gleb Naumenko. Preventing channel jamming. https://blog.bitmex.com/preventing-channel-jamming/. Accessed: 2021-11-03. 5, 15
32. Gleb Naumenko. Mitigating channel jamming with stake certificates for htlc traffic. https://thelab31.xyz/blog/stake-certificates, 2020. 7, 15
33. Cristina Pérez-Solà, Alejandro Ranchal-Pedrosa, Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, and Joaquín García-Alfaro. Lockdown: Balance availability attack against lightning network channels. In *Financial Cryptography*, volume 12059 of *Lecture Notes in Computer Science*, pages 245–263. Springer, 2020. 15
34. Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016. 4
35. B Aditya Prakash, Deepayan Chakrabarti, Nicholas C Valler, Michalis Faloutsos, and Christos Faloutsos. Threshold conditions for arbitrary cascade models on arbitrary networks. *Knowledge and information systems*, 33(3):549–575, 2012. 12
36. Antoine Riard and Gleb Naumenko. Solving channel jamming issue of the lightning network. https://jamming-dev.github.io/book/about.html, 2022. 15
37. Elias Rohrer, Julian Malliaris, and Florian Tschorsch. Discharged payment channels: Quantifying the lightning network's resilience to topology-based attacks. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 347–356. IEEE, 2019. 15

38. Elias Rohrer and Florian Tschorsch. Counting down thunder: Timing attacks on privacy in payment channel networks. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 214–227, 2020. 4, 15
39. Matteo Romiti, Friedhelm Victor, Pedro Moreno-Sanchez, Peter Sebastian Nordholt, Bernhard Haslhofer, and Matteo Maffei. Cross-layer deanonymization methods in the lightning protocol. In *International Conference on Financial Cryptography and Data Security*, pages 187–204. Springer, 2021. 15
40. Rusty Russel. Loop attack with onion routing.. https://lists.linuxfoundation.org/pipermail/lightning-dev/2015-August/000135.html, Aug 2015. 15
41. Rusty Russel. A proposal for up-front payments. https://lists.linuxfoundation.org/pipermail/lightning-dev/2019-November/002275.html, Nov 2019. 15
42. Andrew L Russell. 'rough consensus and running code'and the internet-osi standards war. *IEEE Annals of the History of Computing*, 28(3):48–61, 2006. 3
43. Sergi Delgado Segura. Charges a 1 percent fee to htlc failures. https://github.com/sr-gi/rust-lightning/commit/ce606109d9cbd4217819587b03a4f7a06cd32a13, 2022. 10
44. Anastasios Sidiropoulos and Cosimo Sguanci. Mass exit attacks on the lightning network. *arXiv preprint arXiv:2208.01908*, 2022. 15
45. Stephanie Kaitlyn Skaggs. The financial implications of a cashless society on individual consumers, businesses, banking institutions, and the government. 2014. 7
46. Sergei Tikhomirov, Pedro Moreno-Sanchez, and Matteo Maffei. A quantitative analysis of security, anonymity and scalability for the lightning network. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 387–396. IEEE, 2020. 15
47. Piet Van Mieghem, Jasmina Omic, and Robert Kooij. Virus spread in networks. *IEEE/ACM Transactions On Networking*, 17(1):1–14, 2008. 12
48. Alexei Zamyatin, Mustafa Al-Bassam, Dionysis Zindros, Eleftherios Kokoris-Kogias, Pedro Moreno-Sanchez, Aggelos Kiayias, and William J Knottenbelt. Sok: Communication across distributed ledgers. In *International Conference on Financial Cryptography and Data Security*, pages 3–36. Springer, 2021. 11